JENKINS

Il existe depuis 2008.

**Qu'est-ce que Jenkins?**

Jenkins est un serveur d'automatisation open source autonome qui peut être utilisé pour automatiser toutes sortes de tâches liées à la création, au test, à la livraison ou au déploiement de logiciels.

Jenkins peut être installé via des packages système natifs, Docker, ou même être exécuté de manière autonome par n’importe quel ordinateur sur lequel un environnement d’exécution JRE (Java Runtime Environment) est installé.

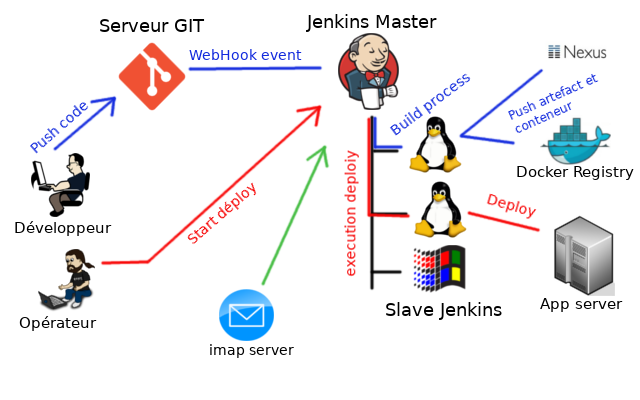
Il est possible de lié Jenkins avec **Subversion** , **Git** , et d'autre contrôleur de révision

Chaque tâche est une job, lors de l'exécution de la job nous exécutons un **build**

Jenkins fonctionne en mode master / slave

**Master** : reçoit les évènements que nous parlions d'évènement généré par le serveur Subversion, git, exécution d'un build manuellement par un utilisateur.

**Slave** : Le serveur slave est donc un agent du master, il réalisera l'opération demandé par le serveur Jenkins. Bien entendu si le serveur Jenkins essaye d'exécuter une commande sur la machine elle doit être installé.



## Installation de Jenkins avec Docker.

Installation de docker sur DEBIAN

1° apt-get install \

apt-transport-https \

ca-certificates \

curl \

gnupg2 \

software-properties-common

2°curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add -

3° add-apt-repository \

"deb [arch=amd64] https://download.docker.com/linux/debian \

$(lsb\_release -cs) \

stable"

4° apt-get update

5° apt-get install docker-ce

Installation de docker Compose

1° Run this command to download the latest version of Docker Compose:

curl -L "https://github.com/docker/compose/releases/download/1.23.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

2° Apply executable permissions to the binary:

chmod +x /usr/local/bin/docker-compose

3° Créer un dossier jenkins + dedans un fichier docker-compose.yml

4° Ecrire ceci dans ce fichier :

version: '2'

services:

jenkins:

image: jenkins/jenkins

container\_name : 'x3-jenkins-f'

hostname: jenkins.train.x3rus.com

environment:

- TZ=Europe/Paris

volumes:

- "/srv/docker/x3-jenkins-f/jenkins-data:/var/jenkins\_home"

ports:

- 8080:8080 # Web interface

- 50000:50000 # Build Executors

**Quelques explications :**

* L'image : Le conteneur de référence original , ici celui fournit par la documentation de Jenkins , donc récupération depuis hub.docker.com
* container\_name : Le nom lors de l'exécution sur le docker host
* hostname : Car on aime avoir un beau hostname ... :P
* environnement : Je définie que le conteneur à le fuseaux horaire de Paris
* Volumes : Comme nous désirons conserver les données de traitement du Jenkins en cas de crash du conteneur ou simplement une mise à jour il est important de le sortir du conteneur !!
* Ports : Ici je ne met aucune redirection de port pour l'adresse IP du docker host , car je n'utiliserai que l'IP du conteneur dans son réseau interne au système . Bien entendu si vous activé le conteneur Jenkins pour être interrogé depuis une autre machine vous devrez mettre en place cette redirection .

5° docker-compose up

6° chown 1000:1000 /srv/docker/x3-jenkins-f/jenkins-data/

7° docker-compose up

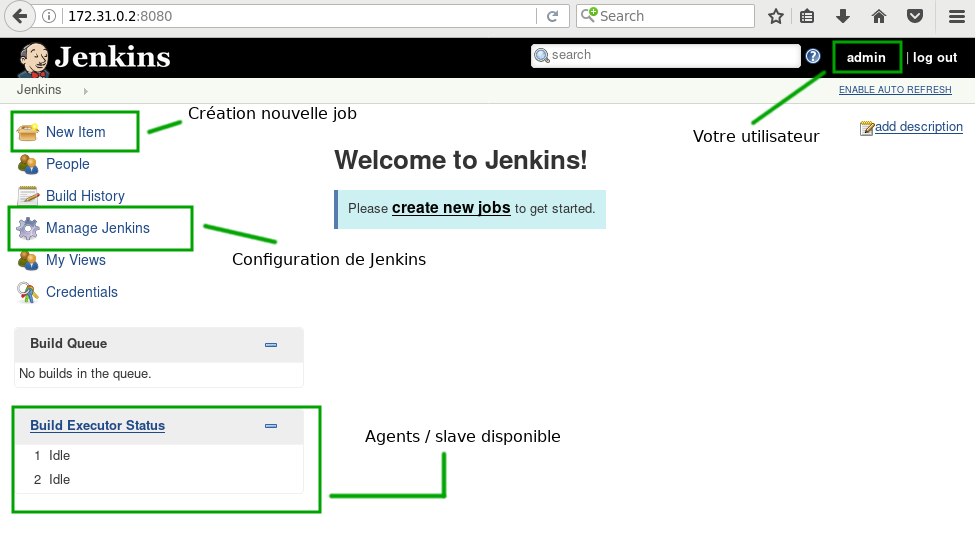
## Quand nous allons sur le service via notre adresse ip :8080 nous arriverons sur la page de garde .

Nous devrons dons saisir la clé qui fut affiché lors du démarrage, dans notre cas : 041688506d8b4f1484ff13ff9c0367ac

Nous installerons les plugins suggérer, si tous ne passent pas on fait réessayer plusiuers fois si nécessaire.

Puis il faut créer le compte admin

On arrive sur cette page d’accueil



* **New Item** : Parfois on appel ça job , parfois item , j'ai voulu le mettre sous le projecteur
* **Manage Jenkins** : Même raison , quand on vient de déployer une application et que l'on veut voir les options de configuration disponible , le voir tous de suite c'est cool :D.
* **Agent / slave** : Telle que mentionné pour le moment nous utiliserons le serveur Jenkins comme **orchestrateur** et serveur d'exécution , mais lorsque nous allons avoir d'autre slave nous aurons une plus grande liste !

On crée une tâche en cliquant sur New itemNous avons donc plusieurs section :

* **General** : Regroupant les informations global de la tâches , description et autre
* **Source code Management** : Il est possible , mais non obligatoire que votre job récupère depuis un dépôt Subversion, git , etc des scripts ou des données.
* **Build Triggers** : Ceci permet d'indiquer qu'est-ce qui déclenchera l'exécution de la tâche , nous y reviendrons
* **Build Environments** : Spécification de l'environnement de build , nous y retrouverons des variables et autre permettant de définir le contexte du build.\* **Build** : La tâche proprement dite , donc les commandes et instruction de la job
* **Post-build Action** : Définition de la tâche quand la job est terminé , nous verrons que nous allons pouvoir par exemple indiquer au système que si le build à bien fonctionné de poussé le résultat dans une voûte ou un docker registry ...

Ce sont les grosses étapes d'une tâche Jenkins , regardons les en détail en réalisant un job. Je vous préviens ne je vais pas couvrir chaque petit option mais celle que je juge importante . De plus je pense que ce sera plus pertinent de réaliser une job et de la modifier en cours de route plutôt que de descendre la page et expliquer le fonctionnement.

Donc allons directement à l'étape de build , vous pouvez cliquer sur l'onglet en haut **build**. Nous allons ajouter une étape , je vais sélectionner **Execute shell**, pour garder ça simple mais honnêtement c'est principalement ce que j'utilise.

Et je vais mettre une petite job simple qui affiche l'heure et le type de CPU de la machine de build



Il suffit de cliquer sur **Build NOW** .

Nous cliquons sur le bouton et maintenant il y a dans la boite **build history** , le build **#1**, il faut sélectionner le build et cliquer sur console pour voir le résultat :

## Intégration avec un contrôleur de révision (GitLab)

La création des tâches dans Jenkins est vraiment bien , comme nous avons pu le constater il est possible d'écrire n'importe quelle script bash ou autre langage. C'est vraiment simple , cependant si vous utilisez uniquement le texte box dans Jenkins vous allez trouver ça long, pourquoi ? Car bien qu'il y est plein de plugin pour Jenkins il n'y a pas d'option de changer le texte box par un éditeur **VIM**.

Voici quelque raison rapidement identifier qui devrez vous décourager vous encourager a utiliser un contrôleur de révision

* Conservation de l'historique des modifications (Qui , Quand, Quoi )
* Utilisation des scripts en dehors de Jenkins
* Segmentation de la tâches en plusieurs script , simplifiant la lecture
* Réutilisation de code dans plusieurs tâches
* Plus simple de réaliser des teste en dehors de Jenkins

Nous avons vu l'utilisation de [GitLab lors de la formation](https://github.com/x3rus/training/blob/master/gitlab/01-presentation.md) sur le sujet, j'utiliserai donc ce système. Le concept est significativement équivalent si vous utilisez **subversion** ou un autre, bien entendu nous avons le concept de branche en plus.

Comme nous avons déjà fait la présentation de gitlab ( Installation et Configuration de base ), je ne couvrirai pas cette partie vous référez à [la formation](https://github.com/x3rus/training/blob/master/gitlab/01-presentation.md) . Nous allons tous de suite passé à l'intégration avec Jenkins , au niveau de Docker puis au niveau applicatif

links:

- gitlab:gitlabsrv

gitlab:

image: 'gitlab/gitlab-ce:latest'

container\_name : 'x3-gitlab-f'

hostname: git.training.x3rus.com

environment:

TZ: Europe/Paris

GITLAB\_OMNIBUS\_CONFIG: |

gitlab\_rails['time\_zone'] = 'Europe/Paris'

gitlab\_rails['gitlab\_email\_from'] = 'noreply@x3rus.com'

gitlab\_rails['manage\_backup\_path'] = true

gitlab\_rails['backup\_path'] = "/var/opt/gitlab/backups"

gitlab\_rails['backup\_archive\_permissions'] = 0640

gitlab\_rails['backup\_keep\_time'] = 604800

gitlab\_rails['smtp\_enable'] = true

volumes:

- '/srv/docker/x3-gitlab-f/gitlab/etc:/etc/gitlab'

- '/srv/docker/x3-gitlab-f/gitlab/logs:/var/log/gitlab'

- '/srv/docker/x3-gitlab-f/gitlab/data/:/var/opt/gitlab'

/srv/docker/gitlab